CHAPTER 2

Deploying ASP.NET Applications in IIS 6.0

Microsoft® Windows® Server 2003 includes support for ASP.NET applications and the Microsoft .NET Framework version 1.1 with the operating system installation. This chapter describes how to deploy ASP.NET applications on a newly installed server running Internet Information Services (IIS) 6.0. Version 1.1 of the .NET Framework is installed with Windows Server 2003. Most ASP.NET applications run without modification on version 1.1 of the .NET Framework.

In This Chapter

Related Information

- For information about improving the availability of your ASP.NET applications, see Ensuring Application Availability in this book.
- For information about ASP.NET-specific considerations when migrating to IIS 6.0, see Migrating IIS Web Sites to IIS 6.0 in this book.
- For information about IIS 6.0 security, see Securing Web Sites and Applications in this book.
- For information about ASP.NET-specific considerations when upgrading to IIS 6.0, see Upgrading an IIS Server to IIS 6.0 in this book.

Overview of Deploying ASP.NET Applications in IIS 6.0

ASP.NET is a unified Web application platform that provides services to help you build and deploy enterprise-class Web applications and XML-based Web services. ASP.NET is supported on the Microsoft® Windows® Server 2003, Standard Edition; Windows® Server 2003, Enterprise Edition; Windows® Server 2003, Datacenter Edition; and Windows® Server 2003, Web Edition operating systems. ASP.NET is installed with the Microsoft .NET Framework version 1.1 as a part of Windows Server 2003. However, to run ASP.NET applications, you must also install IIS 6.0.

Important

ASP.NET is not available on the following operating systems: Microsoft® Windows® XP 64-Bit Edition; the 64-bit version of Windows® Server 2003, Enterprise Edition; and the 64-bit version of Windows® Server 2003, Datacenter Edition. For more information, see "Features unavailable on 64-bit versions of the Windows Server 2003 family" in Help and Support Center for Microsoft® Windows® Server 2003.

The deployment process presented in this chapter describes how to deploy ASP.NET applications on a newly installed IIS 6.0 Web server. Before you begin this process, complete the following steps:

- Install Windows Server 2003, which includes version 1.1 of the .NET Framework, with the default options.
- Install IIS 6.0 with the default settings in **Add or Remove Programs** in Control Panel.

If you need to install ASP.NET applications that were written for IIS 5.0 or version 1.0 of the .NET Framework on a new Web server, see "Migrating IIS Web Sites to IIS 6.0" in this book. If you want to upgrade a Web server running IIS 5.0 that is hosting existing ASP.NET applications, see "Upgrading an IIS Server to IIS 6.0" in this book.

When you configure IIS 6.0 to run in IIS 5.0 isolation mode, the settings in the **<processModel>** section of the Machine.config file are configured in the same way as they were in IIS 5.0 — in the Machine.config or Web.config files. For more information about configuring ASP.NET applications when IIS 6.0 is configured to run in IIS 5.0 isolation mode, see "ASP.NET Configuration" in IIS 6.0 Help, which is accessible from IIS Manager.

Upon completing the process described in this chapter, you will have a Web server running IIS 6.0 and hosting your ASP.NET applications. However, you can further configure the Web server to improve the security and availability of your ASP.NET applications. For more information about configuring your Web server to improve the security and availability of your ASP.NET applications, see "Securing Web Sites and Applications" and "Ensuring Application Availability" in this book.

Note

The configuration settings discussed in this chapter are appropriate for Web sites and applications that are hosted on Web servers on an intranet and the Internet, unless specifically noted.

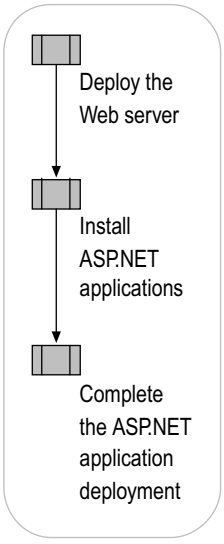
Process for Deploying ASP.NET Applications in IIS 6.0

The process for deploying new ASP.NET applications on a newly installed Web server requires no understanding of earlier versions of IIS or the .NET Framework. All of the ASP.NET configuration sections in the Machine.config and Web.config files are configured the same way in IIS 6.0, except for the **<processModel>** section of the Machine.config file. When IIS 6.0 is configured to run in worker process isolation mode, some of the attributes in the **<processModel>** section of the Machine.config file are now in equivalent IIS 6.0 metabase properties. For more information about how to migrate attributes in the Machine.config file to their equivalent IIS 6.0 metabase property settings, see "Migrating Machine.config Attributes to IIS 6.0 Metabase Property Settings" in "Upgrading an IIS Server to IIS 6.0" in this book.

In addition, if your ASP.NET applications need to retain session state, you must configure IIS 6.0 to use the appropriate ASP.NET application session state method. Depending on the method you select, you might need to configure the ASP.NET state service or Microsoft SQL Server™ to act as the repository for centralized state storage.

The process for deploying ASP.NET applications in IIS 6.0 is shown in Figure 2.1.

Figure 2.1 Deploying ASP.NET Applications in IIS 6.0



Note

Before deploying your ASP.NET applications on a production server, perform the process outlined in this chapter on a test server that is configured identically to your production server.

The following quick-start guide provides a detailed overview of the process for deploying ASP.NET applications in IIS 6.0. You can use this guide to help identify the steps of the ASP.NET application deployment process that you need additional information to complete, and to skip the steps with which you are already

familiar. In addition, all of the procedures that are required to complete the ASP.NET application deployment process are documented in "IIS Deployment Procedures" in this book.

Deploy the Web Server

1. Install Windows Server 2003.
2. Install and configure IIS 6.0.
3. Enable ASP.NET in the Web service extensions list.

Install ASP.NET Applications

1. Create Web sites and virtual directories for each ASP.NET application by doing the following:
   - Create Web sites and home directories.
   - Create virtual directories.
2. Copy ASP.NET application content to the Web server.
3. Enable common storage for ASP.NET session state by completing the following steps:
   - Select the method for maintaining and storing ASP.NET session state.
   - If you decided to maintain session state with the ASP.NET state service, configure out-of-process session state with the ASP.NET state service.
   - If you decided to maintain session state with SQL Server, configure out-of-process session state with SQL Server.
   - Configure encryption and validation keys.
   - Configure ASP.NET to the use the appropriate session state.
   - Secure the ASP.NET session state connection string.

Complete the ASP.NET Application Deployment

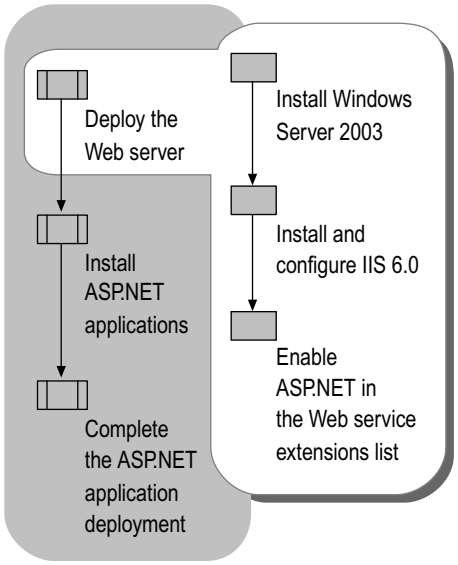
1. Ensure the security and availability of your ASP.NET applications.
2. Verify that the ASP.NET applications were deployed successfully.
3. Back up the Web server.
4. Enable client access to your ASP.NET applications.

Deploying the Web Server

You must install the Web server before you can install your ASP.NET applications. In addition to installing Windows Server 2003, you must install and configure IIS 6.0 on the Web server. You must also enable ASP.NET so that the Web server can run ASP.NET applications.

Figure 2.2 illustrates the process for deploying the Web server.

Figure 2.2 Deploying the Web Server



Installing Windows Server 2003

The deployment process presented here assumes that you install Windows Server 2003 with the default options. If you use other methods for installing and configuring Windows Server 2003, such as unattended setup, your configuration settings might be different.

Note

When you complete the installation of Windows Server 2003, Manage Your Server automatically starts. The deployment process assumes that you quit Manage Your Server, and then further configure the Web server in **Add or Remove Programs** in Control Panel.

For more information about the essential components and services you should enable in Windows Server 2003 see "Enabling Only Essential Windows Server 2003 Components and Services" in "Securing Web Sites and Applications" in this book.

Installing and Configuring IIS 6.0

Because IIS 6.0 is not installed during the default installation of Windows Server 2003, the next step in deploying the Web server is to install and configure IIS 6.0. The deployment process presented here assumes that you install IIS 6.0 with the default options in **Add or Remove Programs** in Control Panel. If you use other methods for installing and configuring Windows Server 2003, such as Manage Your Server, the default configuration settings might be different.

Install and configure IIS 6.0 by completing the following steps:

1. Install IIS 6.0 with only the essential components and services.

   As with installing Windows Server 2003, the primary concern when installing and configuring IIS 6.0 is to ensure that the security of the Web server is maintained. Enabling unnecessary components and services increases the attack surface of the Web server. You can help ensure that the Web server is secure by enabling only the essential components and services in IIS 6.0.

   For more information about how to install IIS 6.0, see "Install IIS 6.0" in "IIS Deployment Procedures" in this book. For more information about the IIS 6.0 components and services that

you should enable, see “Enabling Only Essential IIS Components and Services” in “Securing Web Sites and Applications” in this book.

2. If you want to manage the Web site content by using Microsoft® FrontPage®, install FrontPage 2002 Server Extensions from Microsoft on the Web server.

   For information about how to enable FrontPage Server Extensions after installing them on the Web server, see “Configure Web Service Extensions” in “IIS Deployment Procedures” in this book.

Enabling ASP.NET in the Web Service Extensions List

After you install IIS 6.0, you need to enable ASP.NET. You can enable ASP.NET in **Add or Remove Windows Components**, which is accessible from **Add or Remove Programs** in Control Panel. When you enable ASP.NET by using this method, ASP.NET is also enabled in the Web service extensions list. If you enabled ASP.NET in this way, then you can continue to the next step in the deployment process. To continue to the next step in the deployment process, see “Installing ASP.NET Applications” later in this chapter.

For more information about enabling ASP.NET in **Add or Remove Programs** see “Enable ASP.NET” in “IIS Deployment Procedures” in this book.

ASP.NET might not be enabled in the Web service extensions list if either of the following is true:

- You installed a version, other than version 1.1, of the .NET Framework and ASP.NET from a Web download or as part of an application such as the Microsoft Visual Studio® .NET development tool.
- You disabled ASP.NET in the Web service extensions list because you were not running ASP.NET applications on an existing Web server.

If ASP.NET is not already enabled, view the Web service extensions list in IIS Manager and configure the status of the **ASP.NET v1.1.4322** Web service extension to **Allowed**. For more information about enabling ASP.NET in the Web service extensions list see “Configure Web Service Extensions” in “IIS Deployment Procedures” in this book.

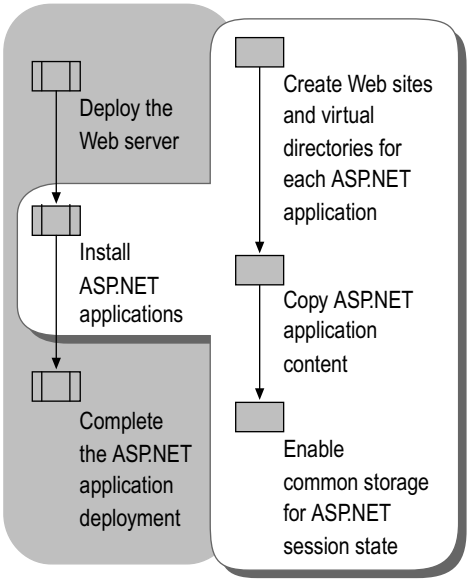
Installing ASP.NET Applications

After the Web server is deployed, you can install your ASP.NET applications. First, you must create a Web site and virtual directories for each ASP.NET application. Then you need to install each ASP.NET application in the corresponding Web site and virtual directory.

When there are provisioning or setup scripts for your ASP.NET applications, use these scripts to install the ASP.NET applications on the Web server. Because the provisioning and setup scripts create the Web sites and virtual directories while installing ASP.NET applications, you do not need to perform any manual steps to install the ASP.NET applications. In this case, run the provisioning or setup scripts to install and configure the Web sites and applications, and then continue to the next step in the application deployment process. To continue to the next step in the process, see “Enabling Common Storage for ASP.NET Session State” later in this chapter.

Figure 2.3 illustrates the process for installing your ASP.NET applications.

Figure 2.3 Installing ASP.NET Applications



Creating Web Sites and Virtual Directories for each ASP.NET Application

For each ASP.NET application, you must create a virtual directory in a new or existing Web site. Later in the installation process, you will install your ASP.NET applications into their corresponding Web sites and virtual directories.

Create the Web sites and virtual directories for your ASP.NET applications by completing the following steps:

1. Create Web sites and home directories.
2. Create virtual directories.

Creating Web Sites and Home Directories

Each Web site must have one home directory. The home directory is the central location for your published Web pages. It contains a home page or index file that serves as a portal to other pages in your Web site. The home directory is mapped to the domain name of the Web site or to the name of the Web server.

Create a Web site and home directory for an ASP.NET application by completing the following steps:

1. Create the folder that will be the home directory for the Web site on the Web server.

   The folder that is the home directory of the Web site contains all of the content and subdirectories for the Web site. The folder can be created on the same computer as the Web server or on a Universal Naming Convention (UNC)–shared folder on a separate server. At a minimum, create the folder on the following:

   - An NTFS file system partition, which helps ensure proper security
   - A disk volume other than the system volume, which reduces the potential of an attack on a Web site bringing down the entire Web server and improves performance

   For more information about securing Web sites and applications, see “Securing Web Sites and Applications” in this book. For more information about creating directories for the Web sites, see “Create a Web Site” in “IIS Deployment Procedures” in this book.

2. Create the Web site on the server.

For more information about how to create a Web site, see "Create a Web Site" in "IIS Deployment Procedures" in this book.

3. If the Web site is FrontPage extended, then configure the Web site on the Web server to be FrontPage extended.

   For more information about how to configure a Web site to be FrontPage extended, see "Configure a Web Site to be FrontPage Extended" in "IIS Deployment Procedures" in this book.

Creating Virtual Directories

A *virtual directory* is a folder name, used in an address, which corresponds to a physical directory on the Web server or a UNC location. This is also sometimes referred to as *URL mapping*. Virtual directories are used to publish Web content from any folder that is not contained in the home directory of the Web site. When clients access content in a virtual directory, the content appears to be in a subdirectory of the home directory, even though it is not.

For security reasons, you might want to move the Web site content to a different disk volume during the application deployment process. You can move the content to another disk volume on the Web server or to a shared folder on a separate server. You can use virtual directories to specify the UNC name for the location where the content is placed, and provide a user name and password for access rights.

For each virtual directory required by the ASP.NET application, create a corresponding virtual directory on the Web server by completing the following steps:

1. Create the folder on the Web server to contain the virtual directory content.

   Ensure that you create the folder in a secure manner that does not compromise the security of the Web server.

   For more information about securing virtual directories, see "Preventing Unauthorized Access to Web Sites and Applications" in "Securing Web Sites and Applications" in this book.

2. Create the virtual directory under the appropriate Web site on the server.

   For more information about how to create virtual directories, see "Create a Virtual Directory" in "IIS Deployment Procedures" in this book.

Copying ASP.NET Application Content

When no installation program or provisioning scripts exist for your ASP.NET application, you can copy the content of the ASP.NET application to the corresponding Web site and virtual directories that you created on the Web server.

You can copy the ASP.NET application content to the Web server by using one of the following methods:

- Run the **Xcopy** command to copy ASP.NET application content to the Web server on an intranet or internal network.
- Use Microsoft Windows Explorer to copy ASP.NET application content to the Web server on an intranet or internal network.
- Use the **Copy Project** command in Visual Studio .NET to copy ASP.NET application content to the Web server on an intranet or internal network, if the application has been developed by using Visual Studio .NET.

Note

FrontPage Server Extensions must be installed on the Web server to use the **Copy Project** command.

- Use the **Publish Web** command in FrontPage to copy ASP.NET application content to the Web server on an intranet or over the Internet, if the Web site that contains the application has been developed by using FrontPage.

  For more information about how to publish ASP.NET application content by using FrontPage, see "Publish Web Site Content with FrontPage" in "IIS Deployment Procedures" in this book.

Enabling Common Storage for ASP.NET Session State

ASP.NET session state lets you share client session data across all of the Web servers in a Web farm or across different worker processes or worker process instances on a single Web server. Clients can access different servers in the Web farm across multiple requests and still have full access to session data.

You can enable common storage for ASP.NET session state by performing the following steps:

1. Select the method for maintaining and storing ASP.NET session state.
2. If you decided to maintain session state with the ASP.NET state service, configure out-of-process session state with the ASP.NET state service.
3. If you decided to maintain session state with SQL Server, configure out-of-process session state with SQL Server.
4. Configure the encryption and validation keys.
5. Configure ASP.NET to use the session state method that you selected in Step 1.
6. Secure the ASP.NET session state connection string in the registry.

Selecting the Method for Maintaining and Storing ASP.NET Session State

You can configure the method used by ASP.NET to maintain session state. ASP.NET supports the following methods for maintaining session state:

- **In-process.** The process for maintaining session state runs in the same worker process as the ASP.NET application and approximates how Active Server Pages (ASP) applications maintain session state. If the worker process for the ASP.NET application recycles, all session state information is lost.
- **Out-of-process.** The process for maintaining session state runs in a different worker process, either on the same Web server or another server, than the worker process for the ASP.NET application. If the worker process for the ASP.NET application recycles, session state information is retained.

You can configure ASP.NET session state settings by modifying the **mode** attribute in the **<sessionState>** section of the Machine.config file for all applications or in the Web.config file for specific applications. The session state settings in the IIS 6.0 metabase only apply to ASP applications. ASP.NET and ASP applications cannot share session state.

Select one of the following methods for storing.ASP.NET session state:

- **In-process, local to the Web server.** ASP.NET session state is managed by and stored on the local Web server. This is the default mode for ASP.NET session state management and it approximates how ASP applications manage session state.

  This method allows the session state to be stored across worker process threads in a Web garden. However, the in-process mode does not allow session state to be store across servers in a Web farm. For more information about Web gardens, see "Configuring Web Gardens" in "Ensuring Application Availability" in this book.

  To use this method to store ASP.NET session state, set the session state **mode** attribute to **InProc**.

- **Out-of-process, by using the ASP.NET state service.** The ASP.NET state service (aspnet_state.exe) runs as a service on Windows Server 2003. You can run the ASP.NET state service local to a Web server to support Web gardens, or on a separate server to support Web farms.

  To use this method to store ASP.NET session state, set the session state **mode** attribute to **StateServer**.

- **Out-of-process, by using a computer running SQL Server.** ASP.NET session state can be managed by and stored in a database on a computer running Microsoft SQL Server. Like the ASP.NET state service, this out-of-process mode provides support for Web gardens or Web farms.

  To use this method to store ASP.NET session state, set the session state **mode** attribute to **SQLServer**.

Table 2.1 compares the methods for maintaining ASP.NET session state, listing the advantages and disadvantages of each method.

Table 2.1 Comparison of Methods for Maintaining ASP.NET Session State

Method	Advantages	Disadvantages
In-process	• Requires no additional computers. • Provides faster access to session state than the out-of-process modes because state is maintained in memory and is in-process.	• Does not provide centralized storage of session state for Web farms. • Provides no redundancy in the event of a Web server failure. • Does not survive application restarts or process recycles.

(continued)

Table 2.1 Comparison of Methods for Maintaining ASP.NET Session State *(continued)*

Method	Advantages	Disadvantages
Out-of-process with the ASP.NET state service running on the local Web server	• Provides centralized storage of session state to support Web gardens. • Requires no additional computers.	• Does not have a mechanism for failover or partitioning. • Out-of-process state requires serializing data and does not perform as well as-in-process state.

		• Provides slower access to session state than in-process mode.
Out-of-process with the ASP.NET state service running on a separate server	• Provides centralized storage of session state to support Web gardens or Web farms.	• Requires an additional computer to store the session state. • Does not have a mechanism for failover or partitioning. • Does not support remote authentication, so administrators must control access to the computer with IPSEC and/or firewall rules. • Out-of-process state requires serializing data and does not perform as well as in-process state. • Provides slower access to session state than in-process mode.
Out-of-process with session state stored on a separate computer running SQL Server	• Provides centralized storage of session state to support Web gardens or Web farms. • Can store session state on a computer running SQL Server that is used for other purposes and share the costs of the server. • Can cluster the computer running SQL Server to provide higher availability.	• Requires an additional computer to store the session state. • Out-of-process state requires serializing data and does not perform as well as in-process state. • Provides slower access to session state than in-process mode.

Note

Before you can use an out-of-process state method for managing and storing session state, you must ensure that the objects defined by and used by your ASP.NET application are *serializable*. Making an object serializable is usually a matter of adding the object class with the [Serializable] attribute. Consult with the ASP.NET application developer to ensure that all of the objects stored in session state by your ASP.NET applications are serializable.

Configuring Out-of-Process Session State with the ASP.NET State Service

If you decide to manage session state by using the ASP.NET state service, you must determine whether you are going to maintain session state for a Web garden or a Web farm. Then you need to ensure that the ASP.NET state service (aspnet_state.exe) is running and, that it is configured to start automatically.

Configure out-of-process session state with the ASP.NET state service by completing one of the following tasks:

- If you are configuring session state for a Web garden, then configure the ASP.NET state service on the local Web server.

  For more information about configuring the ASP.NET state service on the local Web server, see "Configure the State Service on the ASP.NET State Server" in "IIS Deployment Procedures" in this book.

- If you are configuring session state for a Web farm, then complete the following steps:

  1. Install Windows Server 2003 on separate computer that will run the ASP.NET state service.

     The ASP.NET state service is installed as part of the .NET Framework with Windows Server 2003. Install Windows Server 2003 by using the process described in "Installing Windows Server 2003" earlier in this chapter.

  2. Configure the ASP.NET state service on the newly installed Web server.

     For more information about configuring the ASP.NET state service on the new server, see "Configure the State Service on the ASP.NET State Server" in "IIS Deployment Procedures" in this book.

Configuring Out-of-Process Session State with SQL Server

If you decide to store the session state in SQL Server, you need to create the session state database on the computer running SQL Server that is used by the ASP.NET state service. You can create the database by running the InstallSqlState.sql script on the computer running Microsoft SQL Server that is going to be storing the session data. This script creates a database called ASPState, which includes several stored procedures and adds the ASPStateTempApplications and ASPStateTempSessions tables to the TempDB database.

For more information about creating the SQL Server database for storing ASP.NET session state, see "Create a SQL Server Database for Storing ASP.NET Session State" in "IIS Deployment Procedures" in this book.

Configuring Encryption and Validation Keys

ASP.NET uses a key to help protect data so that session state data is only accessible from the Web server that created the data. In a Web garden, all of the worker processes use the same Machine.config or Web.config file, so no additional configuration is necessary. If you are configuring ASP.NET session state for a Web garden, continue to the next step in the deployment process. To continue to the next step in the ASP.NET application deployment process, see "Configuring ASP.NET Applications to Use the Appropriate Session State" later in this chapter.

In a Web farm, each Web server in the farm has a separate Machine.config or Web.config file. As a result, you need to manually configure each Web server in the farm to share the same encryption and validation keys so that they can share session state data. This allows one server to decrypt the session state data created by another server in the Web farm. You can configure the encryption and validation keys in the **<machineKey>** section of the Machine.config file.

Tip

When you want all of the ASP.NET applications on a Web server to use the same encryption and validation keys, make the modifications in the Machine.config file. When you want to customize the encryption and validation keys for each ASP.NET application, modify the corresponding Web.config file for the application.

For each server in the Web farm, identically configure the values of the following attributes in the Machine.config or Web.config file:

- **The validationKey attribute.** This attribute contains the key that is used to validate that data tampering has not occurred. The validation algorithm, which is specified in the **validation** attribute, validates the data by using the key in the **validationKey** attribute. The **validationKey** attribute can range from 40 to 128 hexadecimal characters in length. The default value for the **validationKey** attribute is auto-generated. Configure the value for the **validationKey** attribute to be the same value for all of the servers in a Web farm.
- **The validation attribute.** This attribute is used to configure the validation algorithm used to verify the data. The validation algorithms that you can select include Message Digest 5 (MD5), Secure Hash Algorithm 1 (SHA1), or triple DES (3DES). The Web server validates the data with the key in the **validationKey** attribute and the algorithm specified by the **validation** attribute. Configure the value of the **validation** attribute to be the same value for all of the servers in a Web farm.
- **The decryptionKey attribute.** This attribute contains the key that is used to encrypt data. Valid values are 16 or 48 hexadecimal characters, which corresponds to the Data Encryption Standard (DES) or 3DES algorithm, respectively. The default value for the **decryptionKey** attribute is auto-generated. Configure the value of the **decryptionKey** attribute to be the same value for all of the servers in a Web farm.

For an example that describes how to configure the encryption and validation keys for Microsoft Content Management Server 2002, see the MSDN Online link on the Web Resources page at http://www.microsoft.com/windows/reskits/webresources, and then search for “Best Practices for Authentication for Web Farms”. You can use this example to assist you in configuring the encryption and validation keys for your ASP.NET application.

Configuring ASP.NET Applications to Use the Appropriate Session State

You can configure ASP.NET session state persistence in the **<sessionState>** section of the Machine.config file for all of the ASP.NET applications on the Web server, or you can configure it in the Web.config file for each ASP.NET application. If the **<sessionState>** section of the Web.config file for an ASP.NET application is empty, the session state configuration for the application is inherited from the **<sessionState>** section of the Machine.config file.

If the **<sessionState>** section does not exist in the Machine.config file or the Web.config file, the following is the default session state behavior:

- The session time-out value for an ASP.NET session state is 20 minutes.
- The session state is maintained within all of the applications running in the same application pool.
- The session state is only maintained on the local Web server (where the session state **mode** attribute is set to **InProc**), not across multiple servers in a Web farm.

In addition to the configuration settings that are specific to the mode you selected for maintaining session state, the following attributes need to be configured:

- **The cookieless attribute.** This attribute determines whether or not the session identifier is transferred between the Web server and the client by using cookies. When the **cookieless** attribute is set to **True**, cookies are not used to convey session identifiers. When the **cookieless** attribute is set to **False**, cookies are used to convey session identifiers. The default setting is **False**.
- **The timeout attribute.** This attribute specifies the number of minutes that a session is considered valid. When the time specified in the **timeout** attribute expires, the session is disconnected. The default time-out limit is 20 minutes.

Configure the session state settings in the **<sessionState>** section for the mode that you selected.

Session state is maintained in-process

To maintain session state in-process, you can either delete the **<sessionState>** section or configure the **<sessionState>** section in the Machine.config file or the Web.config file.

The following is an example of the configuration when maintaining session state in-process:

```
<configuration>
  <system.web>
     <sessionState
         mode="InProc"
         cookieless="true"
         timeout="20"
     </sessionState>
  </system.web>
</configuration>
```

Session state is maintained out-of-process with the ASP.NET state service

To configure ASP.NET to maintain session state with the ASP.NET state service, modify the following attributes in addition to the **mode** attribute:

- **The stateConnectionString attribute.** This attribute specifies the IP address and port number where the ASP.NET state service is running. The format for this attribute is "tcpip=*server*:*port*", where *server* is the IP address or host name of the server, and *port* is the TCP port number that the ASP.NET state service is configured to use. The default port number is 42424.
- **The stateNetworkTimeout attribute.** This optional attribute specifies the length of time, in seconds, that the TCP/IP network connection between the Web server and the server running the ASP.NET state service can be idle before the session is abandoned. The default time-out limit is 10 seconds.

The following is an example of the configuration when maintaining session state out-of-process with the ASP.NET state service:

```
<configuration>
  <system.web>
     <sessionState
         mode="StateServer"
         cookieless="true"
         timeout="20"
         stateConnectionString="tcpip=127.0.0.1:42424"
         stateNetworkTimeout="10"
     </sessionState>
  </system.web>
</configuration>
```

Session state is maintained out-of-process with a computer running Microsoft SQL Server

To configure ASP.NET to maintain session state with a computer running SQL Server, modify the **sqlConnectionString** attribute in addition to the **mode** attribute. The **sqlConnectionString** attribute specifies the ODBC data connection string used for establishing the connection to the computer running SQL Server.

The format for the **sqlConnectionString** attribute is "data source=*odbc_connection_string*", where *odbc_connection_string* is any valid ODBC data connection string that is valid for the computer running SQL Server. For more information about creating ODBC data connection strings for Microsoft SQL Server, see the MSDN Online link on the Web Resources page at http://www.microsoft.com/windows/reskits/webresources, and then search for "How to allocate handles and connect to SQL Server (ODBC)".

The following is an example of the configuration when maintaining session state out-of-process with a computer running SQL Server:

```
<configuration>
  <system.web>
     <sessionState
         mode="SQLServer"
         cookieless="true"
         timeout="20"
         sqlConnectionString="data source=localhost;
                              Integrated Security=SSPI;
                              Initial Catalog=northwind"
     </sessionState>
  </system.web>
</configuration>
```

Securing the ASP.NET Session State Connection String

When using either of the out-of-process methods for maintaining session state — by using the ASP.NET state service or Microsoft SQL Server — the ASP.NET session state connection string is stored in the Machine.config or Web.config file in plaintext. You can further secure the ASP.NET session state connection strings by placing the session state connection strings in the registry. Then the Machine.config or Web.config files are modified to point to the corresponding registry keys.

For more information about configuring ASP.NET to store the session connection strings in the registry, see the Microsoft Knowledge Base link on the Web Resources page at http://www.microsoft.com/windows/reskits/webresources, and then search for article 329290, "HOW TO: Use the ASP.NET Utility to Encrypt Credentials and Session State Connection Strings."

Caution

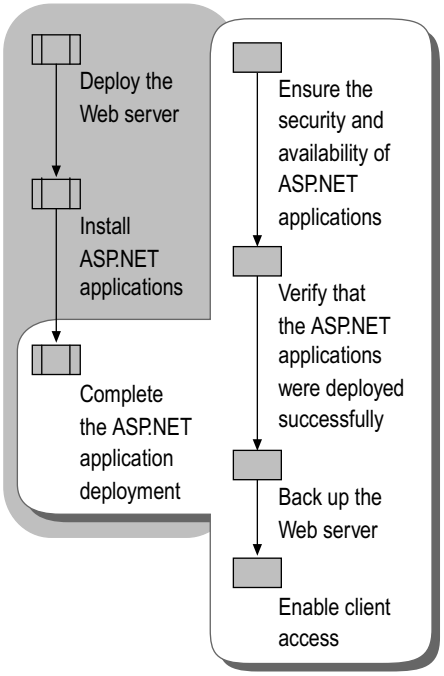
Do not edit the registry unless you have no alternative. The registry editor bypasses standard safeguards, allowing settings that can damage your system, or even require you to reinstall Windows. If you must edit the registry, back it up first and see the Registry Reference on the *Microsoft Windows Server 2003 Deployment Kit* companion CD or on the Web at http://www.microsoft.com/reskit.

Completing the ASP.NET Application Deployment

At this point in the process, your ASP.NET applications are installed on the Web server and the ASP.NET session state settings have been configured on the Web server. Now you need to ensure that the ASP.NET applications are configured to provide the appropriate levels of security and availability for your organizational needs. Then you can verify that the ASP.NET applications have been deployed successfully, capture the current configuration of the Web server, and enable client access to the ASP.NET applications on your Web server. After you complete these last steps, the deployment of your ASP.NET applications is complete.

Figure 2.4 illustrates the process for finalizing the deployment of your ASP.NET applications in IIS 6.0.

Figure 2.4 Completing the ASP.NET Application Deployment Process



Ensuring the Security and Availability of ASP.NET Applications

At this point in the ASP.NET application deployment process, your ASP.NET applications are installed on IIS 6.0, and they are configured to run in the default application pool with the default identity of NetworkService. However, you should take further steps to help ensure that the individual ASP.NET applications are secure. In addition, you can help ensure that if any ASP.NET application fails, the remaining ASP.NET applications remain unaffected.

You can help ensure the security and availability of your ASP.NET applications by completing the following steps:

1. Follow the recommendations and process steps described in “Securing Web Sites and Applications” in this book to help ensure the security of the ASP.NET applications running on your Web server.

2. Follow the recommendations and process steps described in "Ensuring Application Availability" in this book to help ensure the availability of the ASP.NET applications running on your Web server.

Verifying That the ASP.NET Applications Were Deployed Successfully

Before deploying the Web server into a production environment, verify that the ASP.NET applications were deployed successfully by completing the following steps:

1. Review the system log in Windows Server 2003 on the Web server to determine whether any of the ASP.NET applications did not start.

   IIS 6.0 creates entries in the system log when a Web site fails to start for any reason. Search the system log on the Web server to determine whether any errors occurred. For more information about how to troubleshoot Web sites that fail to start, see "Troubleshooting" in IIS 6.0 Help, which is accessible from IIS Manager. For information about how to troubleshoot ASP.NET-specific problems, see "Troubleshooting an ASP.NET Installation" in IIS 6.0 Help, which is accessible from IIS Manager.

2. Perform functional testing of your ASP.NET applications to ensure that they behave as expected.

   You can help eliminate many obvious causes of ASP.NET application failure by reviewing the Windows logs and the application configuration settings. However, the only way to accurately assess the successful deployment of your ASP.NET applications is to perform functional testing. Functional testing is designed to ensure that the ASP.NET applications are functioning correctly for the most common usage scenarios, such as URLs and inputs. This helps ensure that the ASP.NET applications behave as designed, based on typical user interaction.

   Describing the procedures for performing functional testing of your ASP.NET applications is beyond the scope of this chapter. For more information about the general subject of testing, see the MSDN Online link on the Web Resources page at http://www.microsoft.com/windows/reskits/webresources, and then search for "testing".

Backing Up the Web Server

Before you enable client access to the Web server, perform a complete image backup. The purpose of performing this image backup is to provide a point-in-time snapshot of the Web server. If you need to restore the Web server in the event of a failure, you can use this backup to restore the Web server to a known configuration.

Important

Do not continue unless you have a successful backup of the entire Web server. Otherwise, you can lose Web sites, applications, or data that you deployed to the Web server.

For more information about how to back up the Web server, see "Back Up and Restore the Web Server to a File or Tape" in "IIS Deployment Procedures" in this book.

Enabling Client Access

After you have deployed the ASP.NET application to the Web server, you are ready to enable client access to the ASP.NET applications. For a period of time that meets your business needs, monitor the client traffic to ensure that clients are successfully accessing the ASP.NET applications on your Web server and that the clients are experiencing expected response times.

Enable client access to the ASP.NET applications on the Web server by completing the following steps:

1. Create the appropriate DNS entries for the ASP.NET applications running on the Web server.

   For more information about how to create DNS entries for your applications, see "Managing resource records" in Help and Support Center for Windows Server 2003.

2. Monitor client traffic to determine whether clients are successfully accessing the ASP.NET applications.

   For more information about how to monitor client traffic to ASP.NET applications on the Web server, see "Monitor Active Web and FTP Connections" in "IIS Deployment Procedures" in this book.

3. Establish a monitoring period, such as a few hours or a day, to confirm that clients are accessing the ASP.NET applications on the Web server and that they are experiencing response times and application responses that meet or exceed your requirements.

Additional Resources

These resources contain additional information and tools related to this chapter.

Related Information

- "Ensuring Application Availability" in this book for information about configuring your Web server to improve the availability of your ASP.NET applications.
- "IIS Deployment Procedures" in this book for information about specific procedures for deploying ASP.NET applications in IIS 6.0.
- "Migrating Machine.config Attributes to IIS 6.0 Metabase Property Settings" in "Upgrading an IIS Server to IIS 6.0" in this book for information about how to migrate attributes in the Machine.config file to their equivalent IIS 6.0 metabase property settings.
- "Securing Web Sites and Applications" in this book for information about configuring your Web server to improve the security of your ASP.NET applications.
- The Microsoft Knowledge Base link on the Web Resources page at http://www.microsoft.com/windows/reskits/webresources, and then search for article 329290, "How To: Use the ASP.NET Utility to Encrypt Credentials and Session State Connection Strings", for information about configuring ASP.NET to store the session connection strings in the registry.
- The MSDN Online link on the Web Resources page at http://www.microsoft.com/windows/reskits/webresources, and then search for "How to allocate handles and connect to SQL Server (ODBC)", for information about creating ODBC data connection strings for Microsoft SQL Server.
- The MSDN Online link on the Web Resources page at http://www.microsoft.com/windows/reskits/webresources, and then search for "testing", for information about the general subject of testing.

Related IIS 6.0 Help Topics

- "ASP.NET Configuration" in IIS 6.0 Help, which is accessible from IIS Manager, for information about configuring ASP.NET applications when IIS 6.0 is configured to run in IIS 5.0 isolation mode.
- "Troubleshooting" in IIS 6.0 Help, which is accessible from IIS Manager, for information about how to troubleshoot Web sites that fail to start.
- "Troubleshooting an ASP.NET Installation" in IIS 6.0 Help, which is accessible from IIS Manager, for information about how to troubleshoot ASP.NET-specific problems.

Related Windows Server 2003 Help Topics

For best results in identifying Help topics by title, in Help and Support Center, under the **Search** box, click **Set search options**. Under **Help Topics**, select the **Search in title only** check box.

- "Features unavailable on 64-bit versions of the Windows Server 2003 family" in Help and Support Center for Windows Server 2003 for information about features, such as ASP.NET, that are not supported on 64-bit versions of Windows Server 2003.
- "Managing resource records" in Help and Support Center for Windows Server 2003 for information about how to create DNS entries for your applications.